Part III User’s Reference Manual

Chapter 11: *Encrypted vs. Unencrypted Connections*

In this chapter, we provide guidance on determining whether your connection is encrypted, and ensuring that you open an encrypted connection.

Chapter 12: *Kerberos Command Descriptions*

In this chapter we list the native Kerberos commands, and provide a brief description and option list with descriptions adapted from the man pages. Programs that Kerberos provides for ticket and password management include **kinit**, **klist**, **kpasswd** and **kdestroy**.

Chapter 13: *Network Programs Available on Kerberized Machines*

In this chapter we document the Kerberized features of several network programs.

Chapter 11: Encrypted vs. Unencrypted Connections

In this chapter, we provide guidance on determining whether your connection is encrypted, and ensuring that you open an encrypted connection, as needed.

☞

To comply with Fermilab policy, you only strictly need an encrypted network connection when you type your Kerberos password. And to further comply with policy, you should type your Kerberos password over the network extremely rarely, if at all!

11.1 How do you know if your connection is encrypted?

When you’re connecting to a strengthened machine over the network, it’s very important to know if your connection is encrypted. If it is, you can reasonably safely run Kerberos commands that require input of your Kerberos password (see note above). If your connection is not encrypted, you must not type your Kerberos password, since it would be transmitted in the clear. Notice that there are lots of bombs in this chapter!

💣

If you have a chain of multiple connections (e.g., machine1 to machine2, machine2 to machine3, and so on), and if only one connection is unencrypted, then your connection as a whole is **unencrypted**. Do not type your Kerberos password in this case!

Now, we’ll discuss the individual connections ...

11.1.1 Connecting from Kerberized UNIX/Linux Desk-

tops

SSH

If you connect via Kerberized ssh, verify your ssh client configuration to make sure it initiates encrypted sessions. This will vary depending on the ssh client. If you're not sure, use the command with the **`-c`** option as follows:

```
% ssh -c 3des <host>
```

or other argument to `-c` (except for **`none`**).

Other Kerberized Connection Program (e.g., telnet)

Your connection is encrypted if you are connected via one of the Kerberized programs with the "encryption on" flag set. The program generally tells you. For example, for telnet, you can tell if the default is set for encryption by typing the escape character (default is **CTRL-]**, it can be reset with **`-e`** flag), and entering **`status`**. Encryption information should be listed.

For any Kerberized connection program, you can always check the default setting in the `[appdefaults]` section of your `/etc/krb5.conf` file. Look for `encrypt=true` for the program you're using. If encryption is not on by default, use the encryption flag, e.g.,:

```
% rsh -x <host>
```

```
% telnet -x <host>
```

where **`rsh`** and **`telnet`** reside in `/usr/krb5/bin`. Reference Chapter 13: *Network Programs Available on Kerberized Machines* for command syntax.

If you connected in a different way, or if you're not sure, then **assume that the connection is not safe**, log out and log in again as shown here.

11.1.2 Connecting over a CRYPTOCard ssh Session

Verify your ssh client configuration to make sure it initiates encrypted sessions. This will vary depending on the ssh client. If you're not sure, use the command with the **`-c`** option as follows:

```
% ssh -c 3des <host>
```

or other argument to `-c` (except for **`none`**).

11.1.3 Connecting over a CRYPTOCard telnet Session

CRYPTOCard telnet connections are **unencrypted**, and it's **never safe** to issue your Kerberos password. See section 11.2 *If it's unencrypted, what do I do when I need to reauthenticate?*.

11.1.4 Connecting over a CRYPTOCard ftp Session

CRYPTOCard ftp connections are **unencrypted**.

11.1.5 Connecting from an X Terminal

The connection from an X terminal to a host is **never encrypted**, so it's **never safe** to issue your Kerberos password from an X terminal, no matter how secure the connections are beyond that point.

Since X terminals provide no way of encrypting a network connection, we are recommending their replacement by New Internet Computers (NICs). At Fermilab we are providing a CD for configuring the NICs. You will soon be able to get the CDs at the PREP window (for now, send a message to *csi-group@fnal.gov* to request one). For information on the Fermilab configuration, and instructions, go to `http://www-oss.fnal.gov/csi/`, the CD-CSI department's home page, and click on *ThinkNIC*. For the vendor's home page, go to `http://www.thinknic.com/`.

11.1.6 Connecting from a PC Running Windows

Helpful hint: look for the locked lock symbol in your session window to ensure the connection is encrypted!

With ssh

This will vary depending on the ssh client. Verify your client configuration to make sure it initiates encrypted sessions.

With WRQ® telnet client

WRQ® Reflection Security Components v8.0.0 supports ticket forwarding to the remote host, so you shouldn't need to run any commands on the remote system that require password entry. Therefore you may not need an encrypted connection (see section Chapter 19: *Configuring WRQ® Reflection telnet Connections*). If you need to type your password on the remote host for any reason, then you do need an encrypted connection. Make sure that the **WRQ® Reflection** telnet client is configured as described in section 19.6 *Configuring WRQ® Reflection telnet Connections*.:

If you've installed **WRQ® Reflection X**, you can opt to connect to a host directly from the **X CLIENT MANAGER** window, *but* it **does not provide encrypted connections**. If you will need credentials on the host, go through a normal **telnet** connection. **Do not `kinit` from an X window!**

With MIT Kerberos and Exceed 7.0 telnet client

Exceed 7.0 supports ticket forwarding to the remote host, so you shouldn't need to run any commands on the remote system that require password entry. Therefore you may not need an encrypted connection. If you need to type your password on the remote host for any reason, then you do need an encrypted connection. To enable encryption, configure your Kerberized Exceed 7.0 telnet connections as described in section 22.5 *Configuring the Exceed 7 Telnet Application*.

11.1.7 Macintosh: MIT Kerberos and BetterTelnet

In section 24.4 *Configuring Telnet* pay attention to item (3). To summarize:

Invoke **BetterTelnet**. On the **FAVORITES** menu, choose **EDIT FAVORITES**. On the pop-up screen, click **NEW** to create a new configuration or edit an existing one. Change to the **SECURITY** tab, check `Kerberos authentication` and `Kerberos encryption`. Click **OK** to save the configuration.

11.2 If it's unencrypted, what do I do when I need to reauthenticate?

One option for updating tickets on remote sessions is to use the `k5push` script documented in section 9.2.6 *Update Tickets on Remote Terminal Sessions*.

For portal mode connections, a script is provided with the Fermi **kerberos** product as of version v1_2, that safely reauthenticates you on a Kerberized host using your CRYPTOCard over an unencrypted connection. The process exploits the portal mode feature that telnet with a CRYPTOCard always gets you a new key. The script is found at `/usr/krb5/bin/new-portal-ticket` (the script content is provided at the end of this section). Here's how it works:

From your X terminal or unstrengthened machine, you run telnet to a Kerberized machine and use your CRYPTOCard to authenticate. You get logged in on, say, `/dev/ttyp3`, with Kerberos tickets cached in `/tmp/krb5cc_ttyp3`. Now your ticket expires. Still logged into the Kerberized node, you log into the same machine again but using the nonKerberized telnet:

```
% /usr/bin/telnet localhost
```

The machine responds in portal mode, you use your CRYPTOCard, and you’re now logged in on `/dev/ttyp4`, for example, with a new `/tmp/krb5cc_ttyp4` file that has a new cached Kerberos ticket.

So now you have two Kerberos cache files, and you’re logged into the machine twice. One cache file (`/tmp/krb5cc_ttyp3`) has an old expired ticket in it, and the other (`/tmp/krb5cc_ttpy4`) has a fresh, new, usable ticket.

Next, copy your fresh `/tmp/krb5cc_ttyp4` file onto `/tmp/krb5cc_ttyp3` (both cache files live on the destination machine, so you’re doing a safe, local file copy), run **`kdestroy`** (which removes `/tmp/krb5cc_ttyp4`), and log out once. Now you’re back at `ttyp3`, with a fresh new kerberos ticket in `/tmp/krb5cc_ttyp3`, and you can continue doing whatever you were doing when your ticket expired.

Script Contents

```
#!/bin/sh

# get uid
eval ‘id | sed -e ’s/(.*//’‘

# figure ticket cache
if [ "x$KRB5CCNAME" = x ]
then
    krb5file=/tmp/krb5cc_$uid
else
    krb5file=‘echo $KRB5CCNAME | sed -e sxFILE:xx‘
fi

(
   read line
   echo $line
   sleep 1000 &
   pid=$!
   echo ’rkrb5file=‘echo $KRB5CCNAME | sed -e sxFILE:xx‘’
   echo "cp \$rkrb5file $krb5file"
   echo "kdestroy"
   echo "echo xyzzy $pid xyzzy"
   echo "exit"
   wait $pid
) | (
    /usr/krb5/bin/telnet localhost
) |
    while read line
    do
        set : $line
        case $2 in
        Press)
```

```
printf "$line\n"
printf "Enter the displayed response: "
;;
    xyzzy)
kill $3
;;
    esac
done
```

Chapter 12:   Kerberos Command Descriptions

In this chapter we list the native Kerberos commands, and provide a brief description and option list with descriptions adapted from the man pages. Programs that Kerberos provides for ticket and password management include **kinit**, **klist**, **kpasswd** and **kdestroy**.

12.1 kinit

`kinit` obtains and caches a ticket (a ticket-granting ticket, by default) for the default principal or for a specified principal.

12.1.1 Syntax

```
% kinit [-l <lifetime>] [-s <start_time>] [-v] [-p] [-f] [-F] \
  [-k [-t <keytab_file>]] [-r <renewable_life>] [-R] [-a]  \
  [-A] [-c <cache_name>] [-S <service_name>] [<principal>]
```

12.1.2 Option Descriptions

`-l <lifetime>` requests a ticket with the lifetime **`<lifetime>`**. The value for **`<lifetime>`** must be a number followed immediately by a delimiter indicating the unit of time, as follows:

`<n>s` (seconds)

`<n>m` (minutes)

`<n>h` (hours)

`<n>d` (days)

For example: **`kinit -l 90m`**. You cannot mix units; e.g., a value of "**`-l 1h30m`**" will result in an error.

If the **-l** option is not specified, the default ticket lifetime (26 hours, at Fermilab) is used. This option is only useful for specifying a ticket lifetime shorter than the default; to extend the lifetime beyond this limit you must renew the ticket; see **-r** and **-R**.

-s <start_time>

requests a postdated ticket, which can be validated (by action of the user) any time after **<start_time>**. Its lifetime starts when it gets validated. Format for the date and time can be any of the following:

yyyymmddhhmmss

yyyy.mm.dd.hh.mm.ss

yymmddhhmmss

yy.mm.dd.hh.mm.ss

yymmddhhmm

hhmmss

hhmm

hh:mm:ss

hh:mm

Postdated tickets are issued with the "invalid" flag set, and need to be validated before use; see **-v**.

-v	requests that the post-dated ticket in the cache (with the "invalid" flag set) be passed to the KDC for validation. If the start time has passed, the cache is replaced with the validated ticket.
-p	requests proxiable tickets
-f	requests forwardable tickets
-F	requests nonforwardable tickets

-r <renewable_life>

requests renewable tickets, with a maximum lifetime of **<renewable_life>**. If given a value longer than the preconfigured seven day limit, it will be set to seven days. **<renewable_life>** uses the same format as the **<lifetime>** associated with the **-l** option, with the same delimiters.

`-R`	requests renewal of the renewable ticket. Renewal must take place before the ticket's lifetime expires. An expired ticket cannot be renewed, even if the ticket is still within its renewable life.
`-k [-t <keytab_file>]`	requests a host ticket, obtained from a key in the local host's keytab file. The name and location of the keytab file should be specified with the **`-t <keytab_file>`** option; otherwise the default name and location will be used (the default `/etc/krb5.keytab` is not useful here (except to *root*); users cannot read it). Keytab files are generally used for service principals. They are also used for **cron** jobs (see section 10.3.1 *Specific-User Processes (cron Jobs)*).
`-c <cache_name>`	uses **`<cache_name>`** as the credentials (ticket) cache name and location; if this option is not used, the default cache name and location are used. The default credentials cache may vary by system. If the KRB5CCNAME environment variable is set, its value is used to name the default ticket cache. At Fermilab, this variable is typically set to **`FILE:/tmp/krb5cc_<some_string>`**. Any existing contents of the cache are destroyed by **`kinit`**.
`-S <service_name>`	specifies a particular service name to use when getting initial tickets. If this option is not used, you get a ticket-granting-ticket by default.
`-a`	run **`aklog`** after obtaining tickets
`-A`	do not run **`aklog`** after obtaining tickets

12.1.3 Examples

Default

Typically you can run the **`kinit`** command without options. This gets you a 26-hour ticket with the flags `FIA` set by default (`Forwardable`, `Initial`, `Preauthenticated`; flags are viewable using **`klist -f`**, see section 12.2 *klist*), plus an AFS token if AFS is running on the machine.

Get Ticket with Specified Lifetime

Request a ticket valid for three hours using the **`-l`** option:

```
% kinit -l 3h
```

Get Renewable Ticket

Using the **`-r`** option, request a renewable ticket with a maximum renewable lifetime of four days (this sets the `R` flag on the ticket for `Renewable`, and sets the AFS token lifetime to four days):

```
% kinit -r 4d
```

Then, before the lifetime of 26 hours has passed, and before four days expire (you can renew a ticket multiple times within its renewable lifetime, but not after it has expired), renew the ticket using the **`-R`** option:

```
% kinit -R
```

The ticket will remain active an additional 26 hours or until its original four days expires, whichever comes first.

Get Postdated Ticket

Next, request a postdated ticket (using the **`-s`** option), with a lifetime of six hours (the lifetime starts at validation time):

```
% kinit -s 12:25 -l 6h
```

Until it gets validated, the invalid ticket has the flags `FdiIA` set by default, where `d` is `PostDated` and `i` is `Invalid`. Validate it after the start time has passed (using the **`-v`** option):

```
% kinit -v
```

Get Ticket based on Key

The following command requests a TGT for the principal `project/group/host.fnal.gov`, for the duration 30 minutes, with authentication done on the basis of a key previously stored in the keytab file `/usr/tmp/project.keytab` (this command would normally be included in a **cron** job file, not run interactively; see section 10.3.1 *Specific-User Processes (cron Jobs)*):

```
% kinit -l 30m -k -t /usr/tmp/project.keytab \
  project/group/host.fnal.gov
```

If you have an automatic process running as *root*, it is simplest to consider that the host on which the job runs is the party responsible for the accesses it initiates, and have it use the `/etc/krb5.keytab` to obtain credentials as `host/<hostname>.<domain>`:

```
% kinit -l 30m -k host/<hostname>.<domain>
```

12.2 klist

klist lists the Kerberos principal and Kerberos tickets held in a credentials cache (the default), or lists the keys held in a keytab file.

12.2.1 Syntax

```
% klist [-e] [[-c] [-f] [-s] [<cache_name>]] \
  [-k [-t] [-K] [<keytab_name>]]
```

12.2.2 Option/Argument Descriptions

-e	displays the encryption types of the session key and the ticket for each credential in the credential cache, or each key in the keytab file.
-c	lists tickets held in a credentials cache (as opposed to keys in a keytab file). Invalid with **-k**. This is the default if neither **-c** nor **-k** is specified.
-f	shows the flags present in the credentials, using the following abbreviations:

A	preAuthenticated
F	Forwardable
f	forwarded
P	Proxiable
p	proxy
D	postDateable
d	postdated
R	Renewable
I	Initial
i	invalid

Invalid with **-k**.

-s	causes **klist** to run silently (produce no output), while still setting the exit status according to whether it finds the credentials cache. The exit status is “0” if **klist** finds a credentials cache, and “1” if it does not. Invalid with **-k**.

`<cache_name>`	specifies the credentials cache. If **`<cache_name>`** is not specified, **`klist`** will display the credentials in the default credentials cache (unless instructed to operate on a keytab file). If the KRB5CCNAME environment variable is set, its value is used to name the default ticket cache. At Fermilab, this variable is typically set to **`FILE:/tmp/krb5cc_<some_string>`**. Invalid with **`-k`**.
`-k`	lists keys held in a keytab file (as opposed to tickets in a credentials cache). Keytab files are generally used for service principals. Invalid with **`-c`**.
`-t`	displays the time entry timestamps for each keytab entry in the keytab file. Invalid with **`-c`**.
`-K`	displays the value of the encryption key in each keytab entry in the keytab file. Invalid with **`-c`**.
`<keytab_name>`	specifies the keytab file. If **`<keytab_name>`** is not specified, **`klist`** will display the keys in the default keytab file (unless instructed to operate on a credentials cache). Invalid with **`-c`**.

12.2.3 Examples

Most frequently this command is issued with the **`-f`** option to indicate the flags set on each ticket:

```
% klist -f

   Ticket cache: /tmp/krb5cc_ttyp0
   Default principal: aheavey@FNAL.GOV

   Valid starting     Expires            Service principal
   02/11/00       12:45:33              02/12/00       01:45:33
   krbtgt/FNAL.GOV@FNAL.GOV
           Flags: FIA
   02/11/00 12:45:33  02/12/00 01:45:33  afs/fnal.gov@FNAL.GOV
           Flags: FA
```

To list the keys in a keytab file (for example a keytab file created for use with a **cron** job, see section 10.3.1 *Specific-User Processes (cron Jobs)*), use the **`-k`** and **`-t <filename>`** options:

```
% klist -k -t /usr/tmp/user1.keytab

   Keytab name: FILE:/usr/tmp/user1.keytab
   KVNO Timestamp         Principal
```

```
----                                       -----------------
--------------------------------------------------------
   9 02/15/00 10:34:28 user1/cron@FNAL.GOV
```

12.3 kpasswd

The **`kpasswd`** command is used to change a Kerberos principal's password. You can change a principal's password from any account on a machine in the realm. **`kpasswd`** prompts for the current Kerberos password,and if supplied correctly, the user is then prompted twice for the new password, and the password is changed. **`kpasswd`** works even if the old password has expired.

In the FNAL.GOV realm, a policy is in effect that specifies the length and minimum number of character classes required in the new password. The password must be at least ten characters long and contain at least two character classes. For *root*, the password must contain at least 13 characters of at least three classes. The character classes are: lower case, upper case, numbers, punctuation, and all other characters.

12.3.1 Syntax

```
% kpasswd [<principal>]
```

12.3.2 Argument Description

`<principal>`	Change the password for the Kerberos principal **`<principal>`**. If not given, the principal is derived from the identity of the user invoking the **`kpasswd`** command.

12.4 kdestroy

The **`kdestroy`** utility destroys the user's active Kerberos credentials (tickets) by writing zeros to the specified credentials cache that contains them, and then deleting the cache. If the credentials cache is not specified, the default credentials cache specified by $KRB5CCNAME is destroyed.

12.4.1 Syntax

```
% kdestroy [-q] [-c cache_name]
```

12.4.2 Option Descriptions

`-q` Runs quietly. Normally **`kdestroy`** beeps if it fails to destroy the user's tickets. The **`-q`** flag suppresses this behavior.

`-c <cache_name>`

Uses **`<cache_name>`** as the credentials (ticket) cache name and location; if this option is not used, the default cache name and location are used. If the $KRB5CCNAME environment variable is set, its value is used to name the default cache. At Fermilab, this variable is typically set to **`FILE:/tmp/krb5cc_<some_string>`**.

Chapter 13: Network Programs Available on Kerberized Machines

In this chapter we document the Kerberized features of the network connection programs that are usable with Kerberos v5.

13.1 Introduction

The Kerberos V5 network programs are versions of existing UNIX network programs with the Kerberos features added. We call these versions "Kerberized". They include **telnet**, **rsh**, **rlogin**, **FTP**, and **rcp** which come with the installation of a Kerberos 5 client, and **ssh**, **slogin** and **scp** which come with a Kerberized **ssh** client. These programs have the original features of the corresponding non-Kerberized programs, plus additional features that transparently use your Kerberos tickets for negotiating authentication and optional encryption with the remote host. In most cases, all you'll notice is that you no longer have to type your password, because Kerberos has already proven your identity.

☞ Be aware that, depending on how the network program is configured and whether the target machine is Kerberized, you may be prompted for either your login id or password, both, or neither.

You can check the defaults set for the (non-ssh) programs in the `[appdefaults]` section of the `/etc/krb5.conf` file. For **ssh** configuration, see the **ssh** man pages. These defaults can be overridden via command line options (and in the cases of **telnet** and **FTP** when invoked without a hostname argument, via commands inside the program).

In this chapter we list only the command syntax and the Kerberos-added features for these programs.

13.2 Kerberized telnet

Communicate with another host using the TELNET protocol. Use with a host argument to open a connection to that host.

```
% telnet [-8] [-E] [-F] [-K] [-L] [-N] [-S <tos>] \
  [-X <authtype>] [-a] [-c] [-d] [-e <escapechar>] [-f] \
  [-k <REALM>] [-l <user>] [-n <tracefile>] [-r] [-x] \
  [<host> [<port>]]
```

The following are the Kerberos options:

`-a`	attempts automatic login using your tickets. **telnet** will assume you want the same login id on the remote host unless you explicitly specify another (using **`-l`**).
`-f`	forwards a copy of your existing tickets to the remote host, but does not mark them as reforwardable from there. Use of this option overrides any forwarding defaults specified in your machine's configuration files.
`-F`	forwards a copy of your existing tickets to the remote host, and marks them as re-forwardable from there. Use of this option overrides any forwarding defaults specified in your machine's configuration files.
`-k <REALM>`	requests tickets in the specified realm, which may be different from the one the system would use by default.
`-K`	uses your tickets to authenticate to the remote host, but does not log you in; i.e., specifies "no auto-login".
`-N`	turns off ticket forwarding to the remote system. Use of this option overrides any forwarding defaults specified in your machine's configuration files.
`-x`	(encrypt) turns on encryption. Use of this option overrides any encryption defaults specified in your machine's configuration files.
`-X <atype>`	disable **`atype`** type of authentication

Example:

Log in to the remote Kerberized machine fsgi03.fnal.gov, assume your username is different on this machine (**`-l qsmith`**). Forward tickets and mark them as reforwardable from the target machine (**`-F`**):

```
% telnet -F -l qsmith fsgi03.fnal.gov
```

13.3 Kerberized rsh

Connect to a specified host, and execute a specified command on that host.

```
% rsh <host> [-l <login_name>] [-n] [-d] [-k <REALM>] [-f | -F]
  \ [-N] [-x] [-X] <command>
```

If `<command>` is left off, **rsh** runs **rlogin**.

The following are the Kerberos options:

`-f`	forwards a copy of your existing tickets to the remote host, but does not mark them as reforwardable from there. Use of this option overrides any forwarding defaults specified in your machine's configuration files.
`-F`	forwards a copy of your existing tickets to the remote host, and marks them as re-forwardable from there. Use of this option overrides any forwarding defaults specified in your machine's configuration files.
`-k <REALM>`	requests tickets in the specified realm, which may be different from the one the system would use by default.
`-n`	This is not a Kerberos option, but we include it with a usage note. As in non-Kerberized rsh, **`-n`** redirects input from the special device `/dev/null`. If you put a command **`rsh <host> <command>`** in the background with **`&`**, it will stop because only a foreground process can access the tty for input. If you make it **`rsh -n <host> <command>`**, the **`rsh`** command does not have the tty open for input at all, so it does not get stopped.
`-N`	turns off ticket forwarding to the remote system. Use of this option overrides any forwarding defaults specified in your machine's configuration files.
`-x`	(encrypt) turns ON encryption for the session Use of this option overrides any encryption defaults specified in your machine's configuration files.
`-X`	turns OFF encryption of the session. Use of this option overrides any encryption defaults specified in your machine's configuration files.

Example:

Run the command **`date`** on the remote Kerberized machine fsui03.fnal.gov, and assume your username is different on it (**`-l qsmith`**). The command doesn’t require Kerberos tickets in order to run, nor does it require encryption (**`-X`** turns it off):

```
% rsh fsgi03.fnal.gov -l qsmith -X date
```

13.4 Kerberized rlogin

Log into a remote host. Kerberos authentication is used in place of the rhosts mechanism to determine if user is authorized to use remote account.

```
% rlogin <rhost> [-e<c>] [-8] [-c] [ -a] [-f] [-F] [-N]    \
  [-t <termtype>] [-n] [-7] [-noflow] [-d] [-k <REALM>] [-x]\
  [-X] [-L] [-l <username>]
```

The following are the Kerberos options:

`-f`	forwards a copy of your existing tickets to the remote host, but does not mark them as reforwardable from there. Use of this option overrides any forwarding defaults specified in your machine’s configuration files.
`-F`	forwards a copy of your existing tickets to the remote host, and marks them as re-forwardable from there. Use of this option overrides any forwarding defaults specified in your machine’s configuration files.
`-k <REALM>`	requests tickets in the specified realm, which may be different from the one the system would use by default.
`-N`	turns off ticket forwarding to the remote system. Use of this option overrides any forwarding defaults specified in your machine’s configuration files.
`-x`	(encrypt) turns ON encryption for the session Use of this option overrides any encryption defaults specified in your machine’s configuration files.
`-X`	turns OFF encryption of the session. Use of this option overrides any encryption defaults specified in your machine’s configuration files.

Example:

Log into the remote Kerberized machine fsui03.fnal.gov, assume your username is different on it (**`-l qsmith`**), forward a reforwardable copy of the local Kerberos credentials (**`-F`**):

```
% rlogin fsgi03.fnal.gov -l qsmith -F
```

13.5 Kerberized FTP

Transfer files to and from a remote host. FTP prompts the user for a command. Type **`help`** to see a list of commands.

```
% ftp [-v] [-d] [-i] [-n] [-g] [-k <REALM>] [-f] [-x] [-u] [-t]\
  [<host>]
```

The following are the Kerberos options:

`-f`	requests that your tickets be forwarded to the remote host. (This is necessary if the remote host runs AFS.)
☞ **`-k <REALM>`**	Ignore this option of the ftp client. It has nothing to do with Kerberos v5. It does work for telnet and the r-commands.
`-n`	no auto-login attempt at initial connection, but still does Kerberos authentication
`protect <level>`	(issued at the **`ftp>`** prompt) sets the protection level. The level **`clear`** is “no protection”; **`safe`** ensures data integrity, and **`private`** encrypts the data and ensures data integrity.
`-u`	restrains **FTP** from attempting auto-authentication; also disables auto-login.

Note: If your local and remote login names don’t match, you can enter your login name for the remote system at the prompt that you get after you issue the **`ftp`** command.

Examples:

Transfer files from a remote nonKerberized machine www.xyz.org, and assume your username is different on it:

```
% ftp www.xyz.org
```

```
Connected to xyz.org.
220 ...
500 AUTH not understood.
KERBEROS_V4 rejected as an authentication type
Name (www.xyz.org:aheavey): anneh
331 Password required for anneh.
Password:
230 User anneh logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
-rw-rw-r--   1 batavia  site23       1700 Jan 25 10:52 header_1.GIF
...
ftp> get header_1.GIF
```

```
local: header_1.GIF remote: header_1.GIF
200 PORT command successful.
150 Opening BINARY mode data connection for header_1.GIF (1700 bytes).
226 Transfer complete.
1700 bytes received in 0.016 seconds (1e+02 Kbytes/s)
ftp> bye
221 Goodbye.
```

Transfer files from a remote Kerberized machine abc.minos-soudan.org that runs AFS (you must forward credentials, **`-f`**). Assume your username is different on each machine. Set the protection to “private” in order to encrypt the data and ensure data integrity:

`% ftp -f abc.minos-soudan.org`

```
Connected to abc.minos-soudan.org.
...
220 abc.minos-soudan.org FTP server (Version 5.60) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
Name (abc.minos-soudan.org:aheavey): crluser
232 GSSAPI user aheavey@FNAL.GOV is authorized as crluser
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> private
200 Data channel protection level set to private.
ftp> get lmnop.qrs
...
ftp> bye
221 Goodbye.
```

13.6 Kerberized rcp

Copy files between machines. Each file or directory argument is either a remote file name of the form `remote_host:path` or a local file name/path.

```
% rcp [-p] [-x] [-X] [-k <REALM>] [-D <port>] [-n] [-F] [-N] \
  [-c <cache>] [-C <config>] <file1> <file2>
```

or

```
% rcp [-p] [-x] [-X] [-k <REALM>] [-r] [-D <port>] [-n] [-F]\
  [-N] [-c <cache>] [-C <config>]   <file> ... <directory>
```

The following are the Kerberos options:

`-c <cache>`	uses credentials file **`<cache>`** instead of default
`-F`	forwards credentials to remote system (This is needed if the other end runs AFS.)
`-k <REALM>`	requests tickets for the remote host in the specified realm, which may be different from the one the system would use by default.
`-N`	turns off ticket forwarding to the remote system. Use of this option overrides any forwarding specified in your machine's configuration files.
`-x`	(encrypt) turns on encryption.
`-X`	turns off encryption of the session. Use of this option overrides any encryption specified in your machine's configuration files.

Examples:

Copy the local files `def.histo` and `ghi.histo` to your home directory on the remote machine jkl.myuniv.edu. Assume the remote machine does not run AFS. Your username is the same on both:

```
% rcp def.histo ghi.histo jkl.myuniv.edu:
```

Copy the local directory `histo` and all subdirectories to your home directory on the remote machine jkl.myuniv.edu. Assume the remote machine does not run AFS. Your username is the same on both:

```
% rcp /path/to/histo jkl.myuniv.edu:
```

Copy all the files from the directory `/path/to/mno` on the remote node pqr.myuniv.edu into the local directory `~stu/vwx` (quote the first argument to prevent filename expansion from occurring on the local machine):

```
% rcp “pqr.myuniv.edu:/path/to/mno/*” ~stu/vwx
```

13.7 Kerberized su (ksu)

The discussion here is adapted from the **ksu** man pages. See them for more information, in particular for option descriptions. The command syntax is:

```
% ksu [<target_user>] [-n <target_principal_name>] [-c        \
  <source_cache_name>] [-C <target_cache_name>] [-k] [-D] [-r \
  <time>] [-pf] [-l <lifetime>] [-zZ] [-q] [-e <command>      \
  [<args ...>]] [-a [<args ...>]]
```

The Kerberos V5 **ksu** program is a Kerberized version of the **su** program that has two missions: one is to securely change the real and effective user ID to that of the target user, the other is to create a new security context.

To fulfill the first mission, **ksu** operates in two phases: authentication and authorization. Resolving the target principal name is the first step in authentication. If the source user is *root* or the target user is the source user, no authentication or authorization takes place. In all other cases, **ksu** looks for an appropriate Kerberos ticket in the source cache. If no ticket is in the cache, then depending on how **ksu** was compiled, the user may be prompted for a Kerberos password.

Make sure you are logged in using an encrypted connection before typing your password!

Upon successful authentication, **ksu** checks whether the target principal is authorized to access the target account. In the target user's home directory, authorization is based on whether appropriate entries exist in either `.k5login` or `.k5users`, or by name-mapping rules if neither file exists.

ksu can be used to create a new security context for the target program. The target program inherits a set of credentials from the source user. By default, this set includes all of the credentials in the source cache plus any additional credentials obtained during authentication. The source user is able to limit the credentials in this set.

13.8 Kerberized ssh and slogin

The **`ssh`** and **`slogin`** commands are intended to replace **`rsh`** and **`rlogin`** (see sections 13.3 *Kerberized rsh* and 13.4 *Kerberized rlogin*) and to provide secure encrypted connections between two untrusted hosts over an insecure network. If the **`<command>`** argument is left off, **`ssh`** runs **`slogin`**.

```
% ssh [-a] [-c idea|blowfish|des|3des|arcfour|none]     \
  [-e <escape_char>] [-i <identity_file>] [-l <login_name>]\
  [-n] [-k] [-V] [-o <option>] [-p <port>] [-q] [-P] [-t] [-v]\
  [-x] [-C] [-g] [-L <port>:<host>:<hostport>] [-R \
  <port>:<host>:<hostport>] <hostname> [<command>]
```

`-c`	Specifies cipher for encrypting connection; not needed if specified in configuration file
`-k`	Disables forwarding of the kerberos tickets. This may also be specified on a per-host basis in the configuration file.

Any Kerberos options would be used within **`-o <ssh-options>`**.

Examples:

From your local machine, log into the remote node fsgi03.fnal.gov on which your (different) username is qsmith. Respond **`yes`** if asked if you want to continue:

```
% slogin fsgi03.fnal.gov -l qsmith
   Host key not found from the list of known hosts.
   Are you sure you want to continue connecting (yes/no)?
```

From your local machine, run the **`date`** command on the remote node fsgi03.fnal.gov, but don't start a session:

```
% ssh fsgi03.fnal.gov -l qsmith date
```

13.9 Kerberized scp

Copy files between hosts on a network, using ssh for data transfer.

```
% scp [-a] [-A] [-q] [-Q] [-p] [-r] [-v] [-B] [-C] [-L] [-1] \
  [-S <path_to_ssh>] [-o <ssh-options>][-P <port>]    \
  [-c idea|blowfish|des|3des|arcfour|none]  [-i <identity>]\
  [[user@host1:]filename1... [user@host2:]filename2
```

Any Kerberos options would be used within **`-o <ssh-options>`**.

Example:

Log into a Kerberized machine at Fermilab, and pull files from a remote machine, mynode.myuniv.edu. On the remote node the username is qsmith, and on the local node, it's quentins. The user wants to pull a file from mynode.myuniv.edu to his local Fermilab machine:

```
% scp qsmith@mynode.myuniv.edu:/home/qsmith/muonrun47.histo \
  ~quentins/geant4/work/muonhistos
```